Optimal Timing Control of Switched Systems With Applications to Optimal Bridge Repairs

A Thesis
Presented to
The Academic Faculty

by

Johan Isaksson

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2006

Optimal Timing Control of Switched Systems With Applications to Optimal Bridge Repairs

Approved by:

Dr. Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yorai Wardi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 4th, 2006

ACKNOWLEDGEMENTS

I would like to thank Dr. Magnus Egerstedt, Assistant Professor in the Systems and Controls group at Georgia Institute of Technology for supervising me and giving me the opportunity to conduct this work in the Georgia Robotics and Intelligent Systems Laboratory (GRITS Lab.). Another thank you to PhD Candidate Henrik Axelsson in the GRITS Lab. for helpful discussions and insightful advice. I would also like to thank my supervisor and examiner at Chalmers Institute of Technology, Dr. Bengt Lennartsson, professor in the department of Signals and Systems. My last thank you goes to professor A. Bayen in Civil and Environmental Engineering at University of California at Berkeley for providing the motivation problem to this thesis.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF FIGURES . . . . . **v**

SUMMARY . . . . . **vi**

I INTRODUCTION **1**

II PROBLEM FORMULATION AND GRADIENT SOLUTION **4**

III ALGORITHM **10**

3.1 Steepest Descent Algorithm with Armijo Stepsizes 10

3.2 Gradient-Projection Algorithm 13

3.2.1 Feasible Descent Direction Algorithm 13

3.2.2 Armijo Stepsize 14

3.3 Steepest Descent Gradient-Projection Algorithm with Armijo Stepsizes . . 16

IV MATLAB IMPLEMENTATION **17**

V OPTIMIZING THE REPAIRS ON A BRIDGE **18**

5.1 Bridge Model 18

5.1.1 Solution 20

VI LIMITATIONS **24**

VII CONCLUSION **25**

APPENDIX A — MATLAB IMPLEMENTATION **26**

APPENDIX B — GLOBAL OPTIMUM CODE **33**

REFERENCES **34**

LIST OF FIGURES

1 Visualization of a switched dynamical system 2
2 Example 2.1 state trajectory 8
3 Example 3.1 gradient descent parameters 12
4 Example 3.1 state trajectories 12
5 Example 3.2 gradient descent parameters 15
6 Example 3.2 state trajectory 16
7 The bridge model 18
8 Bridge state trajectory 21
9 Bridge gradient descent parameters 22
10 Cost dependency of number of repairs 23

SUMMARY

Following results over recent years, this thesis enhances the problem of minimizing a cost functional defined on a state trajectory of an autonomous switched dynamical system. The cost functional traditionally used, is augmented with explicit costs on the switching times and the final time is set by a constraint as opposed to being given. An equation for the gradient of the cost functional is derived and an algorithm is proposed for computing local minima. The algorithm is based on existing steepest descent methods including the Armijo procedure and gradient projection. A matlab implementation of the algorithm is developed in order to solve optimal problems that can be modelled with costs on or between the switching times. An existing problem, the motivation for this research, where repairs on a bridge is to be optimized, is provided and solved.

CHAPTER I

INTRODUCTION

A system that consist of a combination of continuous and discrete events is called a hybrid system. Such systems arise in a variety of applications when a continuous and/or discrete time process is interfaced with logics or decision-making. Models for these systems was early proposed, [22] and [20], but following rapid development in the area more recently, fairly general theory for hybrid systems was discussed in [13] with a particular attention to process control. Lately, [6] established a basic framework with general theorems on stability analysis of hybrid systems while also concluding that the content of the hybrid systems area need a lot of further exploring .

Within the area of optimal control theory and especially hybrid systems lies a special set of problems concerning dynamical switched systems. The issue is still, as of being an optimal control problem, to minimize a cost functional on the trajectory of the system subject to certain constraints, while the dynamics of the system changes between different modes. This thesis handles such problems where existing theory needs to be enhanced to be able to take on problems with special characteristics not before treated.

Consider a dynamical switched system where the dynamical response changes according to a sequence of predefined control laws. The state trajectory is denoted by $\{x(t)\}_{t=t_0}^{T}, \quad x \in \mathbb{R}^n$ and its dynamics are given by $\dot{x} = f(x,t)$, where $f : \mathbb{R}^n \times [t_0, T] \to \mathbb{R}^n$ consists of a sequence of functions $f_i : \mathbb{R}^n \to \mathbb{R}^n, \quad i = 1, \ldots N+1$. Let the times where the dynamic response of the system changes from $\dot{x} = f_i$ to $\dot{x} = f_{i+1}$ be denoted by $\tau_i, \quad i = 1, \ldots, N$. Denote the vector containing switching times by $\bar{\tau} \in \mathbb{R}^N$. Define $\tau_0 := t_0$, $\tau_{N+1} := T$ and let the sequence of dynamical functions be fixed so that the transition times become subject to the following constraint:

$$t_0 = \tau_0 \leq \tau_1 \leq \ldots \leq \tau_N \leq \tau_{N+1} = T. \tag{1}$$

Hence, the dynamics of the system can be summarized as

$$\dot{x} = f_i(x(t)), \quad t \in [\tau_{i-1}, \tau_i), \quad i = 1, \ldots, N+1. \tag{2}$$

Note that the state trajectory is then well defined and continuous on $[t_0, T]$. Furthermore, let the initial condition for the state trajectory be given and fixed, $x(t_0) = x_0 \in \mathbb{R}^n$. The dynamical system can be visualized as in figure 1.

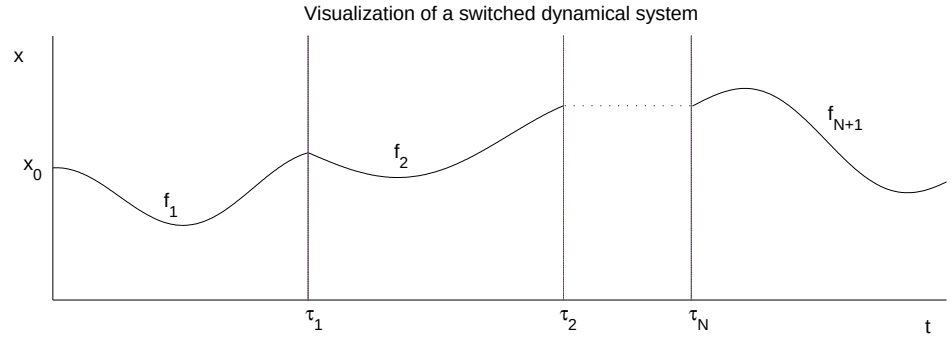

Figure 1: The state of a switched dynamical system, where the dynamics change at the different switching times, τ_i.

Such systems are seen, among others, in applications where a control module has to switch its attention between different subsystems [14], [17] and [21] or when the issue is to collect data sequentially from different sensor sources [5], [7] and [12].

Optimal control problems concerning switched systems where the control variable consist of not only a proper switching law but also an input function $u(t)$ has been of big interest lately [4], [11], [15], [18], [19] and [25]. [4] propose a general framework for hybrid systems, [19] formulate a class of optimal control problems for general hybrid systems with nonlinear dynamics and [25] present a new approach for solving optimal control problems for switched systems.

Autonomous systems, without a present $u(t)$, is investigated in [10], which is restricted to the case where the subsystems are linear, [23] and [24] consider general nonlinear systems while [9] takes it further by developing a more simple formula for the gradient of the cost functional of the state in terms of the switching times.

Define on the trajectory a cost functional

$$J(\bar{\tau}) = \int_{t_0}^{T} L(x(t))dt \tag{3}$$

for a given continuously differentiable cost function $L : \mathbb{R}^n \to \mathbb{R}$ so that the problem, denoted by P_σ becomes

$$\min_{\bar{\tau}} J(\bar{\tau})$$

s.t.

$$\begin{cases} \dot{x} = f_i(x(t)), & t \in [\tau_{i-1}, \tau_i) \\ x(t_0) = x_0 \end{cases}$$

and

$$t_0 = \tau_0 \leq \tau_1 \leq \ldots \leq \tau_N \leq \tau_{N+1} = T \tag{4}$$

as in [23], [9] and [2]. Note that the system considered is autonomous and so the control parameters consist only of the switching times $\tau_1 \ldots \tau_N$. Hence the problem consists in determining the optimal transition times between the different dynamical functions.

Efficient descent-based numerical algorithms for such systems have already been produced in [23] and [9]. [8] suggest a possible extension concerning the number of switches as well as the switching sequence being a part of the variable parameter. This thesis aims to extend the autonomous problem with a fixed dynamical sequence, P_σ, as follows:

- Augment $J(\bar{\tau})$ with explicit costs on switching times, $g(\tau_i)$, and/or times between switches, $g(\tau_i - \tau_{i-1})$

- Have the final time T given by a constraint, $T = p(\bar{\tau})$

The rest of this thesis is organized as follows: a formula is developed for the gradient of the cost functional after defining the costate in section 2. Section 3 presents an algorithm based on previous research suitable for this problem and section 4 briefly describes a matlab implementation of it. Section 5 provides an numerical solution to the bridge repairs problem that motivated this work. The problem concerns optimizing the repairs on a bridge subject to specific costs and certain constraints. The problem is solved with the implementation of the algorithm. Section 6 discuss limitations to the developed theory and section 7 concludes the thesis.

CHAPTER II

PROBLEM FORMULATION AND GRADIENT SOLUTION

The will to examine the autonomous switching problem with the mentioned extensions comes from wanting to penalize having two switches close together and to not explicitly knowing the final time but in some sense know when it occurs. Both of the extension are needed to solve the bridge repairs optimization problem properly. The cost functional, eq 3 is through desired extensions turned into

$$J = \int_{t_0}^{p(\bar{\tau})} L(x)dt + \sum_{j=1}^{N+1} g_j(z_j) \tag{5}$$

where

$$z_j = \tau_j - \tau_{j-1}$$

The problem is described only for costs on the times between switches which can easily be turned into explicit costs on the transition times by replacing z_i with τ_i .

Proposition 2.1 *Define the costate, $\lambda(t) \in \mathbb{R}^n$, by the backwards differential equation*

$$\begin{cases} \lambda(p(\bar{\tau})) = 0 \\ \dot{\lambda} = -\left(\frac{\partial L}{\partial x}(x)\right)^T - \left(\frac{\partial f_{i+1}}{\partial x}(x)\right)^T \lambda \end{cases}$$
$$t \in (\tau_i, \tau_{i+1}] \quad i = N, N-1, \ldots, 0. \tag{6}$$

Then, the following equation is valid for all $i \in \{1 \ldots N\}$

$$\frac{dJ}{d\tau_i} = \lambda^T(\tau_i)\big[f_i(x(\tau_i)) - f_{i+1}(x(\tau_i))\big] + \frac{\partial p}{\partial \tau_i} L\big(x(p(\bar{\tau}))\big) + \frac{\partial g_i}{\partial z_i} - \frac{\partial g_{i+1}}{\partial z_{i+1}} \tag{7}$$

Proof 2.1 *The proof for proposition 2.1 is here made with a Calculus of Variations approach by perturbing τ_i.*

$$J = \int_{t_0}^{T} L(x)dt + \sum_{j=1}^{N+1} g_j(z_j)$$

$$\begin{cases} z_j = \tau_j - \tau_{j-1} \\ T = p(\bar{\tau}) \end{cases}$$

$$J(\bar{\tau}) = \int_{t_0}^{p(\bar{\tau})} \big(L(x) + \lambda^T[f(x) - \dot{x}]\big)dt + \sum_{j=1}^{N+1} g_j(z_j)$$

$$= \int_{t_0}^{\tau_1} \big(L(x) + \lambda^T[f_1(x) - \dot{x}]\big)dt + \dots$$
$$+ \int_{\tau_N}^{p(\bar{\tau})} \big(L(x) + \lambda^T[f_{N+1}(x) - \dot{x}]\big)dt + \sum_{j=1}^{N+1} g_j(z_j)$$

$$\tau_i \to \tau_i + \epsilon\theta_i \qquad \Rightarrow \qquad \begin{cases} \bar{\tau} \to \bar{\tau} + \epsilon\bar{\theta} \\ x \to x + \epsilon\eta \\ z_i \to z_i + \epsilon\theta_i \\ z_{i+1} \to z_{i+1} - \epsilon\theta_i \end{cases}$$

$$J(\bar{\tau} + \epsilon\bar{\theta}) = \int_{t_0}^{p(\bar{\tau}+\epsilon\bar{\theta})} \big(L(x + \epsilon\eta) + \lambda^T[f(x + \epsilon\eta) - \dot{x} - \epsilon\dot{\eta}]\big)dt$$
$$+ \sum_{j \neq i, i+1} g_j(z_j) + g_i(z_i + \epsilon\theta_i) + g_{i+1}(z_{i+1} - \epsilon\theta_i)$$

$$= \int_{t_0}^{\tau_1} \big(L(x + \epsilon\eta) + \lambda^T[f_1(x + \epsilon\eta) - \dot{x} - \epsilon\dot{\eta}]\big)dt + \dots$$
$$+ \int_{\tau_{i-1}}^{\tau_i + \epsilon\theta_i} \big(L(x + \epsilon\eta) + \lambda^T[f_i(x + \epsilon\eta) - \dot{x} - \epsilon\dot{\eta}]\big)dt$$
$$+ \int_{\tau_i + \epsilon\theta_i}^{\tau_{i+1}} \big(L(x + \epsilon\eta) + \lambda^T[f_{i+1}(x + \epsilon\eta) - \dot{x} - \epsilon\dot{\eta}]\big)dt + \dots$$
$$+ \int_{\tau_N}^{p(\bar{\tau}+\epsilon\bar{\theta})} \big(L(x + \epsilon\eta) + \lambda^T[f_{N+1}(x + \epsilon\eta) - \dot{x} - \epsilon\dot{\eta}]\big)dt$$
$$+ \sum_{j \neq i, i+1} g_j(z_j) + g_i(z_i + \epsilon\theta_i) + g_{i+1}(z_{i+1} - \epsilon\theta_i)$$

$$= \left\{ \begin{array}{ll} \eta(t) = 0, & t \in [0, \tau_i) \\ \dot{\eta}(t) = 0, & t \in [0, \tau_i) \end{array} \right\}$$

$$\begin{aligned} = & \int_{t_0}^{\tau_i} \big(L(x) + \lambda^T [f(x) - \dot{x}] \big) dt \\ & + \int_{\tau_i}^{\tau_i + \epsilon\theta_i} \big(L(x) + \epsilon \frac{\partial L}{\partial x} \eta + \lambda^T [f_i(x) + \epsilon \frac{\partial f_i}{\partial x} \eta - \dot{x} - \epsilon\dot{\eta}] \big) dt + \dots \\ & + \int_{\tau_N}^{p(\bar{\tau} + \epsilon\bar{\theta})} \big(L(x) + \epsilon \frac{\partial L}{\partial x} \eta + \lambda^T [f_{N+1}(x) + \epsilon \frac{\partial f_{N+1}}{\partial x} \eta - \dot{x} - \epsilon\dot{\eta}] \big) dt \\ & + \sum_{j \neq i, i+1} g_j(z_j) + g_i(z_i) + \epsilon \frac{\partial g_i}{\partial z_i} \theta_i + g_{i+1}(z_{i+1}) - \epsilon \frac{\partial g_{i+1}}{\partial z_{i+1}} \theta_i + o(\epsilon) \end{aligned}$$

$$\begin{aligned} \delta J(\bar{\tau}, \bar{\theta}) = & \lim_{\epsilon \to 0} \frac{J(\bar{\tau} + \epsilon\bar{\theta}) - J(\bar{\tau})}{\epsilon} = \{\eta(t) \to 0, t \in [\tau_i, \tau_i + \epsilon\theta_i)\} \\ & = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left\{ \int_{\tau_i}^{\tau_i + \epsilon\theta_i} \lambda^T [f_i - f_{i+1}] dt \right. \\ & \quad + \int_{\tau_i + \epsilon\theta_i}^{\tau_{i+1}} \big(\epsilon \frac{\partial L}{\partial x} \eta + \lambda^T [\epsilon \frac{\partial f_{i+1}}{\partial x} \eta - \epsilon\dot{\eta}] \big) dt + \dots \\ & \quad + \int_{\tau_N}^{p(\bar{\tau})} \big(\epsilon \frac{\partial L}{\partial x} \eta + \lambda^T [\epsilon \frac{\partial f_{N+1}}{\partial x} \eta - \epsilon\dot{\eta}] \big) dt \\ & \quad \left. + \int_{p(\bar{\tau})}^{\epsilon \frac{\partial p}{\partial \tau_i} \theta_i} L(x) dt + \epsilon \frac{\partial g_i}{\partial z_i} \theta_i - \epsilon \frac{\partial g_{i+1}}{\partial z_{i+1}} \theta_i + o(\epsilon) \right\} \end{aligned}$$

$$= \{ \textit{Mean-Value Theorem, Integration by Parts} \}$$

$$\begin{aligned} = & \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left\{ \epsilon \lambda(\tau_i) \left[f_i(\tau_i) - f_{i+1}(\tau_i) \right] \theta_i \right. \\ & + \int_{\tau_i + \epsilon\theta_i}^{\tau_{i+1}} \epsilon\eta \left[\frac{\partial L}{\partial x} + \lambda^T \frac{\partial f_{i+1}}{\partial x} - \dot{\lambda}^T \right] dt + \dots \\ & + \int_{\tau_N}^{p(\bar{\tau})} \epsilon\eta \left[\frac{\partial L}{\partial x} + \lambda^T \frac{\partial f_{N+1}}{\partial x} - \dot{\lambda}^T \right] dt + \epsilon \frac{\partial p}{\partial \tau_i} L \left(x(p(\bar{\tau})) \right) \theta_i \\ & \left. - \epsilon \lambda(p(\bar{\tau})) \eta(p(\bar{\tau})) + \epsilon \lambda(\tau_i) \eta(\tau_i) + \epsilon \frac{\partial g_i}{\partial z_i} \theta_i - \epsilon \frac{\partial g_{i+1}}{\partial z_{i+1}} \theta_i + o(\epsilon) \right\} \end{aligned}$$

$$= \Big\{ \eta(\tau_i) = 0 \Big\}$$

$$
\begin{aligned}
&= \lambda(\tau_i)\left[f_i(\tau_i) - f_{i+1}(\tau_i)\right]\theta_i + \int_{\tau_i}^{\tau_{i+1}} \eta\left[\frac{\partial L}{\partial x} + \lambda^T \frac{\partial f_{i+1}}{\partial x} - \dot{\lambda}^T\right] dt + \dots \\
&\quad + \int_{\tau_N}^{p(\bar{\tau})} \eta\left[\frac{\partial L}{\partial x} + \lambda^T \frac{\partial f_{N+1}}{\partial x} - \dot{\lambda}^T\right] dt + \frac{\partial p}{\partial \tau_i} L\left(p(\bar{\tau})\right)\theta_i \\
&\quad - \lambda(p(\bar{\tau}))\eta(p(\bar{\tau})) + \frac{\partial g_i}{\partial z_i}\theta_i - \frac{\partial g_{i+1}}{\partial z_{i+1}}\theta_i \\
&= \frac{dJ}{d\tau_i}\theta_i
\end{aligned}
$$

choose

$$
\begin{cases}
\lambda^T(p(\bar{\tau})) = 0 \\
\dot{\lambda}^T = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f_{j+1}}{\partial x}, \quad t \in [\tau_j, \tau_{j+1}), \quad j = i \dots N
\end{cases}
$$

$$
\Rightarrow \quad \boxed{\frac{dJ}{d\tau_i} = \lambda(\tau_i)\left[f_i(\tau_i) - f_{i+1}(\tau_i)\right] + \frac{\partial h}{\partial \tau_i} L\left(p(\bar{\tau})\right) + \frac{\partial g_i}{\partial z_i} - \frac{\partial g_{i+1}}{\partial z_{i+1}}}
$$

■

The equation for the costate is, despite the problem's extensions, defined just as in [9]. The resulting equation for the partial derivative is similar to the one derived in [9] but differs due to the augmentations of the cost functional with one term corresponding to the final time constraint and an additional two per switch, one for the time since the former switch and one for the time to the next.

Example 2.1 (Cost functional gradient) *Consider a one dimensional problem with only one switch, depicted in figure 2. Its dynamics is given by*

$$
\dot{x} = f(t) = \begin{cases} f_1 = 1 & t \in [0, \tau) \\ f_2 = -x & t \in [\tau, T) \end{cases}
$$

Let the cost functions be $L(x) = \frac{1}{2}\|x\|^2 = x^2$ *and* $g(z) = \frac{(z-2)^2}{2}$. *Furthermore, let the final time be given by* $x(T) = \frac{1}{2}$ *and the initial point* $x_0 = 1$. *The cost functional for the problem*

is then expressed as

$$J = \frac{1}{2}\int_0^{p(\tau)} x^2 dt + \frac{(z_1-2)^2}{2} + \frac{(z_2-2)^2}{2}$$

$$z_1 = \tau - 0 \qquad z_2 = T - \tau$$

Calculate the gradient for $\tau = 0.5$.

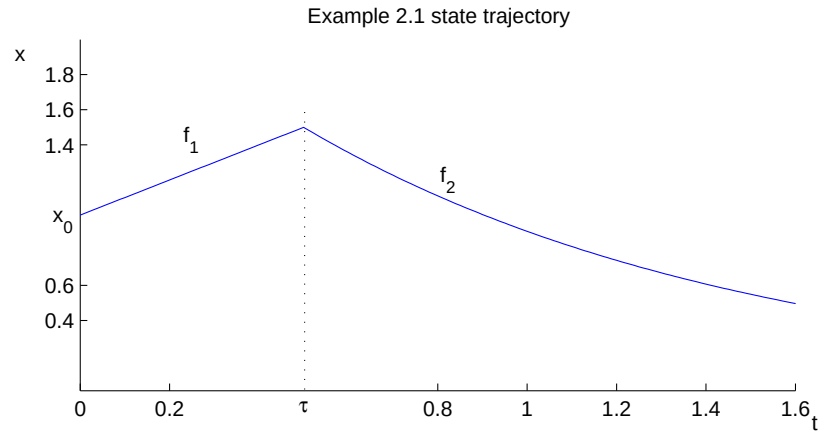

Figure 2: The state of a one-dimensional system where the dynamics change at time τ.

Solution 2.1 *First, an explicit expression for* $T = p(\tau)$ *is needed. To do this, calculate the solution to the differential equation for* x*:*

$$x = \begin{cases} t + x_0 & t \in [0, \tau_1) \\ e^{-(t-\tau)}(\tau + x_0) & t \in [\tau_1, T) \end{cases}$$

$$x(T) = \frac{1}{2} \quad \Rightarrow \quad T = p(\tau) = \tau - \ln\frac{x(T)}{\tau + x_0} \approx 1.6$$

The differential equation for the costate also has to be solved:

$$\frac{\partial L}{\partial x} = x$$
$$\frac{\partial f_2}{\partial x} = -1$$

$$\Rightarrow \quad \begin{cases} \lambda(T) = 0 \\ \dot{\lambda} = -x + \lambda \end{cases}$$

$$\Rightarrow \quad \lambda = \frac{-x}{e^T}e^t + x$$

To express the cost functional derivative, a few more partial derivatives are needed

$$\begin{aligned}
\frac{\partial p}{\partial \tau} &= \frac{1}{\tau + x_0} + 1 \\
\frac{\partial g}{\partial z_1} &= z_1 - 2 = \tau - 2 \\
\frac{\partial g}{\partial z_2} &= z_2 - 2 = T - \tau - 2
\end{aligned}$$

Then calculate the derivative of the cost functional

$$\begin{aligned}
\frac{dJ}{d\tau} &= \lambda(\tau)\big[f_1(x(\tau)) - f_2(x(\tau))\big] + \frac{\partial p}{\partial \tau} L\big(x(p(\tau))\big) + \frac{\partial g_1}{\partial z_1} - \frac{\partial g_2}{\partial z_2} \\
&= \left(\frac{-x(\tau)}{e^T} e^\tau + x(\tau)\right)[1 + x(\tau)] \\
&\quad + \left(\frac{1}{\tau + x_0} + 1\right) L(x(T)) + (\tau - 2) - (T - \tau - 2) \\
&= \left(\frac{-\tau - x_0}{e^T} e^\tau + \tau + x_0\right)[1 + \tau + x_0] + \left(\frac{1}{\tau + x_0} + 1\right) 0.125 + 2\tau - T \\
&\approx 2.1
\end{aligned}$$

The derivative is positive and hence, the optimal switching time is before the actual.

□

CHAPTER III

ALGORITHM

The similarity with the results in [9] makes it appropriate to use the algorithm there derived. A steepest descent algorithm [16] with Armijo stepsizes [1] was proposed. The algorithm is enhanced with a gradient-projection algorithm [2] to make the solution be part of the feasible set according to the constraint in eq. 1.

As discussed in [9], the simple structure of the partial derivatives $\frac{dJ}{d\tau_i}, \quad i = 1, \ldots, N$ and the fact that the same costate is used for all of them makes the computation of the partial derivatives easier. The algorithm proposed follows, where i represents the iteration index:

3.1 Steepest Descent Algorithm with Armijo Stepsizes

Parameters: $\alpha \in (0, 1)$ and $\beta \in (0, 1)$

1. compute $h(\bar{\tau}_i) = -\nabla J(\bar{\tau}_i)$
2. compute $k = \min\{k \geq 0 \mid J(\bar{\tau}_i + \beta^k h(\bar{\tau}_i)) - J(\bar{\tau}_i) \leq -\alpha\beta^k \|h(\bar{\tau}_i)\|^2\}$
3. set $\bar{\tau}_{i+1} = \bar{\tau}_i + \beta^k h(\bar{\tau}_i)$

Next, a numerical example to show the feasibility of the algorithm.

Example 3.1 (Steepest descent algorithm) *Consider a two-dimensional system with two switches that evolves over the time interval* $[0, 1]$*. Let the cost functional defined on the system trajectory be*

$$J = \frac{1}{2} \int_0^1 \|x\|^2 dt$$

and the linear dynamics of the system

$$\dot{x} = f(x) = \begin{cases} f_1 = \begin{bmatrix} 2 & -1 \\ -1 & -1 \end{bmatrix} x & t \in [0, \tau_1) \\ f_2 = \begin{bmatrix} -1 & 1 \\ 2 & 1 \end{bmatrix} x & t \in [\tau_1, \tau_2) \\ f_3 = \begin{bmatrix} 2 & 0 \\ 1 & -1 \end{bmatrix} x & t \in [\tau_2, 1) \end{cases}$$

Find locally optimum switching times for a starting vector

$$x_0 = \begin{bmatrix} 0.5 \\ 0.9 \end{bmatrix}$$

Solution 3.1 *Use the Armijo parameters* $\alpha = \beta = 0.5$ *and chose a starting switching vector, for example*

$$\bar{\tau} = [0.3 \ 0.7]$$

Using the Steepest Descent Algorithm with Armijo Stepsizes that terminates when discretized time steps are larger than wanted steps, a local minima is found at the switching times

$$\bar{\tau} = [0.63 \ 0.91]$$

Figure 3 shows the gradient descent parameters for every iteration and figure 4 displays the initial and final trajectories for the two dimensions of the state.

□

This algorithm was proven to be globally convergent to stationary points [16]. Generally, α and β are both set to $\alpha = \beta = 0.5$. Next, the gradient-projection algorithm in [2] proposed to make the following changes to the steepest descent algorithm with Armijo stepsizes in order to reach a feasible solution subject to the constraint in eq. 1.

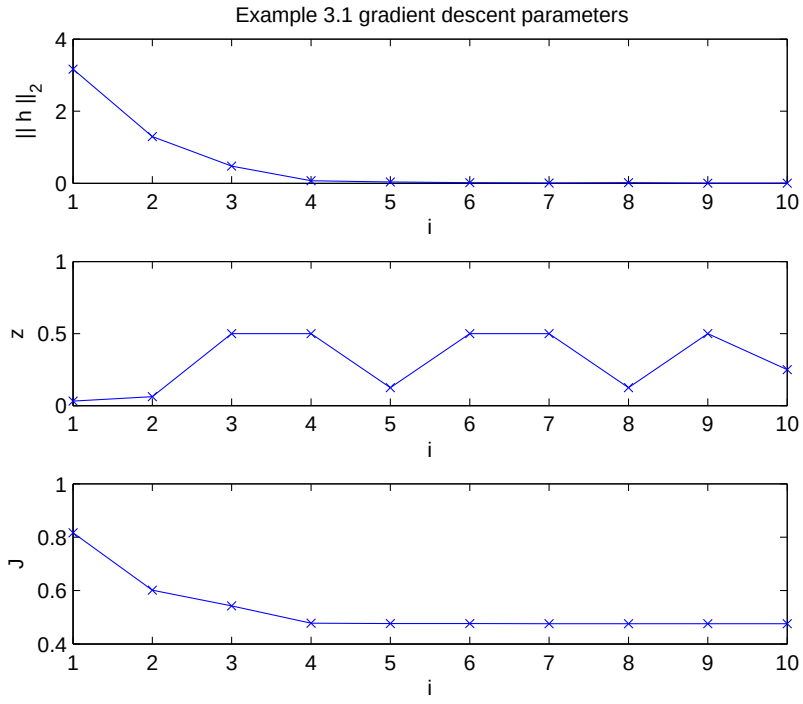

Figure 3: Gradient descent parameters for every iteration i: Top: Norm of the gradient, $\|h\|$. Middle: Stepsize, $z = \beta^k$. Bottom: Cost, J.

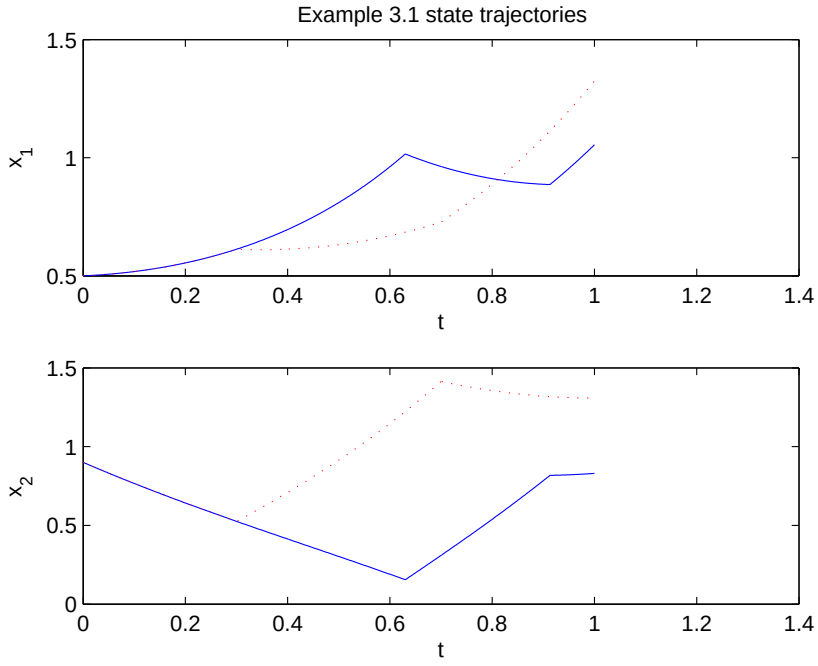

Figure 4: Initial (dotted) and final trajectories for the two dimensional state, x_1 and x_2.

3.2 Gradient-Projection Algorithm

Denote the feasible set defined by eq. 1 as

$$\Phi = \{\bar{\tau} \in \mathbb{R}^N \mid t_0 \leq \tau_1 \leq \ldots \leq \tau_N \leq T\}. \tag{8}$$

Furthermore, denote the set of feasible directions from a point $\bar{\tau}$ as

$$\Psi(\bar{\tau}) = \{h(\bar{\tau}) \in \mathbb{R}^N \mid \exists \tilde{\zeta} > 0, \forall \zeta \in [0, \tilde{\zeta}),\ \bar{\tau} + \zeta h(\bar{\tau}) \in \Phi\} \tag{9}$$

where ζ denotes the Armijo step size. Define for a fixed $\bar{\tau} \in \Phi$ a *block* as a contigous integer-set $\{k, \ldots, m\} \subset \{1, \ldots, N\}$ such that $\tau_i = \tau_k \quad \forall i \in \{k, \ldots, n\}$. Let a block be *maximal* if no subset thereof is a block. For a fixed maximal block $\{k, \ldots, n\}$, define

$$r_{l,i} = \frac{1}{i-l+1} \sum_{j=l}^{i} \frac{dJ(\bar{\tau})}{d\tau_j}$$
$$\forall l \in \{k, \ldots, n\}, \quad \forall i \in \{l, \ldots, n\}. \tag{10}$$

Now, apply the following two steps to the steepest descent algorithm, different computation of the descent direction and computation of the stepsize:

3.2.1 Feasible Descent Direction Algorithm

Compute the feasible descent direction $h(\bar{\tau}_i)$:

1. Set $l = k$

2. Compute $r_{max} = \max\{r_{l,i} \mid i = l, \ldots, n\}$

   Set $m = \max\{i = l, \ldots, n : r_{l,i} = r_{max}\}$

3. $\forall i \in \{l, \ldots, m\}, h_i(\bar{\tau}) = r_{max}$ unless either case is true:

   - $\bar{\tau}_m = t_0$ and $r_{max} > 0$
   - $\bar{\tau}_m = T$ and $r_{max} < 0$

   If so, set $h_i(\bar{\tau}) = 0$

4. If $m = n$, exit. Else, set $l = m + 1$ and go to 2)

The vector $h(\bar{\tau})$ computed is proven to be the projection of $-\nabla J(\bar{\tau}_i)$ onto Ψ.

3.2.2 Armijo Stepsize

Compute the Armijo stepsize ζ by

$$\begin{aligned} k = \min\{k \geq 0 \mid \bar{\tau} + \beta^k h(\bar{\tau}) \in \Phi, \\ J(\bar{\tau} - \beta^k h(\bar{\tau})) - J(\bar{\tau}) \leq \alpha\beta^k < h(\bar{\tau}), \nabla J(\bar{\tau}) >\} \in \mathbb{N} \\ \zeta = \beta^k \end{aligned} \tag{11}$$

To illustrate the feasibility of the algorithms enhancements, a numerical example is provided next.

Example 3.2 (Gradient-projection algorithm) *Consider a one-dimensional system on the time interval* $[0, 1]$ *with seven initial switches*

$$\bar{\tau} = [0.1\ 0.2\ 0.3\ 0.6\ 0.7\ 0.8\ 0.85]$$

and an initial starting point $x_0 = 0.8$. *Assume it is wanted to keep the trajectory close to* $x = 1$, *i.e. the cost functional is*

$$J = \int_0^1 (x-1)^2 dt$$

The system evolves according to the following nonlinear dynamics:

$$\dot{x} = f(x) = \begin{cases} f_1 = -x^2 - 2 & t \in [0, \tau_1) \\ f_2 = x & t \in [\tau_1, \tau_2) \\ f_3 = x^2 + 2 & t \in [\tau_2, \tau_3) \\ f_4 = -x & t \in [\tau_3, \tau_4) \\ f_5 = -x^2 - 2 & t \in [\tau_4, \tau_5) \\ f_6 = -x & t \in [\tau_5, \tau_6) \\ f_7 = x & t \in [\tau_6, \tau_7) \\ f_8 = -x & t \in [\tau_7, T) \end{cases} \tag{12}$$

The system is subject to the constraint

$$0 \leq \tau_1 \leq \ldots \leq \tau_7 \leq 1$$

Find a switching vector that yields a local cost minima.

Solution 3.2 *Appending the gradient projection to the gradient descent algorithm and using the Armijo constants $\alpha = \beta = 0.5$, a local minima is by simulation found at*

$$\bar{\tau} = [0\ 0.25\ 0.31\ 0.66\ 0.66\ 0.68\ 0.9]$$

Note that the algorithm now keeps the constraint, with $\tau_0 = 0$, and also eliminates one switch by having τ_4 and τ_5 equal. Figure 5 shows the gradient descent parameters and figure 6 shows the state trajectory. Note that the norm of the gradient at a point increases, which is a result of a discretization of the switching times.

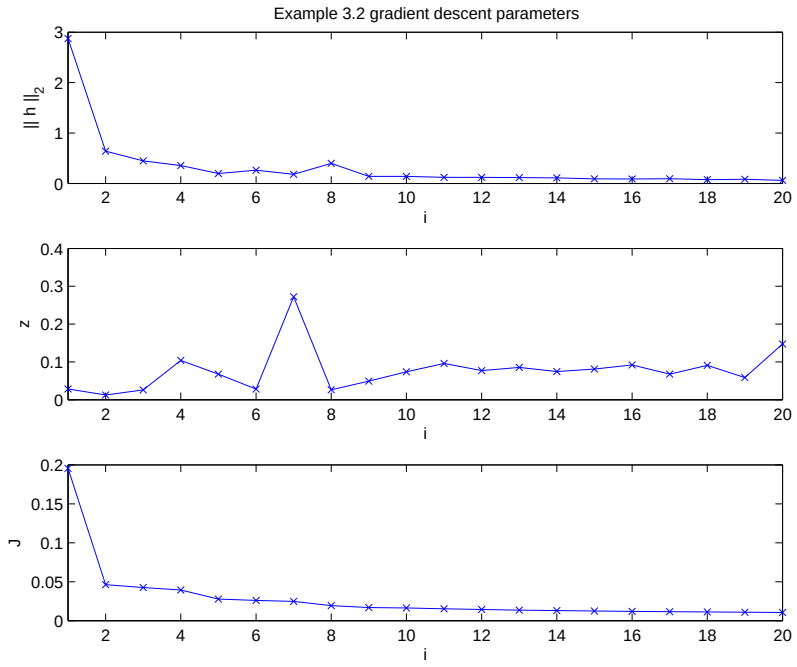

Figure 5: Gradient descent parameters for every iteration i: Top: Norm of the gradient, $\|h\|$. Middle: Stepsize, $z = \beta^k$. Bottom: Cost, J.

□

Applying the gradient-projection to the steepest descent algorithm can be summarized to the following algorithm, suitable for the problem concerned in this thesis. i represents iteration index:

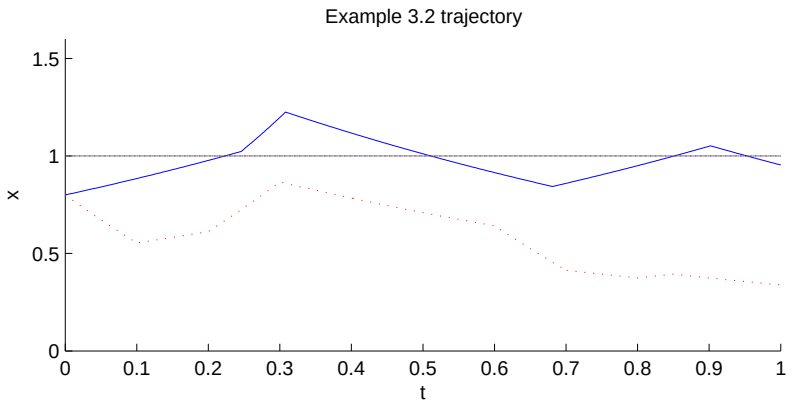

Figure 6: Initial (dotted) and final trajectory for the one dimensional state, x_1.

3.3 Steepest Descent Gradient-Projection Algorithm with Armijo Stepsizes

1. choose a valid starting vector $\bar{\tau}$
2. solve $x(t)$ forwards from eq. 2 and a given x_0
3. solve $\lambda(t)$ backwards from eq. 6
4. compute $\nabla J(\bar{\tau}_i)$ from eq. 7
5. compute $h(\bar{\tau}_i)$ according to the feasible descent direction algorithm
6. compute ζ_i from eq. 11
7. set $\bar{\tau}_{i+1} = \bar{\tau}_i + \zeta_i h(\bar{\tau}_i)$ and go to 1)

Next section uses algorithm 3.3 in a matlab implementation.

CHAPTER IV

MATLAB IMPLEMENTATION

The steepest descent gradient-projection algorithm with Armijo stepsizes was implemented in Matlab, in order to achieve numerical solutions to problem P_σ. The implementation, code found in appendix A, is based on seven matlab m-files - six functions and one script. The main script file, **switch_cost.m** executes algorithm 3.3 and calls for the six functions:

const.m Takes a vector $\bar{\tau}$ as input and verifies if it meets the specified constraint, eq 1. Returns a 1 or a 0.

dJdtao.m Returns the vector $\nabla J(\bar{\tau}_i)$ from eq. 7 for the partial derivatives.

dlambdadt.m Returns $\dot{\lambda}$ depending on eq. 6 for the costate for a given mode, x and λ.

f.m Returns the dynamics, eq. 2, i.e. $f_i(x)$, for a given mode and value x.

J.m Evaluates the cost functional in eq. 5 for a given vector $\bar{\tau}$.

lambdaT.m Returns $\lambda(T)$ as defined in eq 6.

Note that all the partial derivatives in eq. 6 and eq. 7 must be calculated and entered into the different function files.

The code was also extended to take account for the dual dynamic functions associated with each switch in the example bridge problem provided in the next section. The code in appendix A is provided with the equations from that problem.

CHAPTER V

OPTIMIZING THE REPAIRS ON A BRIDGE

The problem that provided motivation for the research in this thesis concerns a bridge, to have its reparations optimized. The model, provided by Professor A. Bayen at UCB [3], is presented in fig.7. The x-axis represents the bridge quality and the time instants where the slope changes is related to repairs. As seen in the figure, a repair does not increase the quality of the bridge, it only slows down the breaking-down process for a certain time, t_2, after which the reduction of quality increases again. Also, the quality is constant for a time t_1, at quality level x_0 after the bridge has been raised.

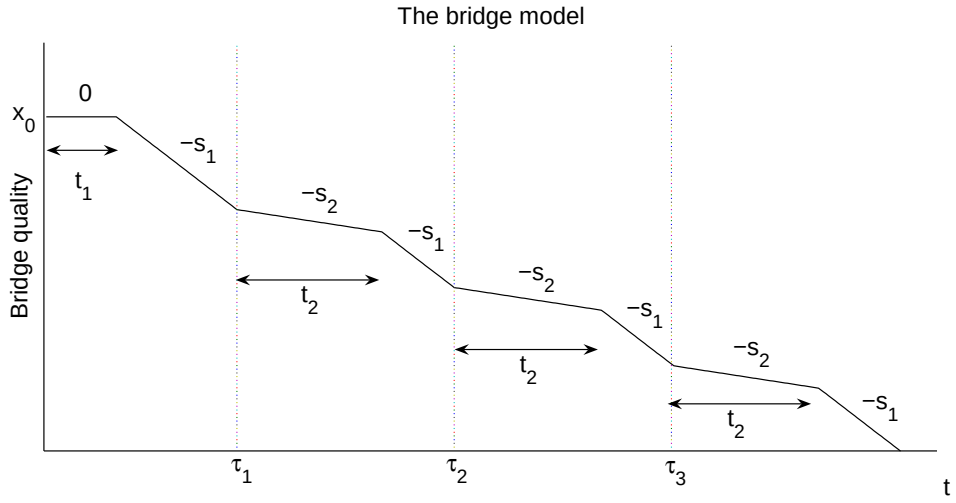

Figure 7: Bridge repair optimization: The quality degeneration of the bridge as a function of time and repairs

5.1 Bridge Model

The problem can be modelled as an autonomous linear switched system where the state trajectory $x(t) \in \mathbb{R}$ represents the bridge quality and the switching times represent repairs on the bridge. The final time T is explicitly given by $T = p(\bar{\tau})$. The systems dynamics, for

n switches, is described as

$$\dot{x} = f(t) = \begin{cases} f_1 = 0 & t \in [0, t_1) \\ f_2 = -s_1 & t \in [t_1, \tau_1) \\ f_3 = -s_2 & t \in [\tau_1, \tau_1 + t_2) \\ f_4 = -s_1 & t \in [\tau_1 + t_2, \tau_2) \\ f_5 = -s_2 & t \in [\tau_2, \tau_2 + t_2) \\ \vdots & \\ f_{2N+1} = -s_2 & t \in [\tau_N, \tau_N + t_2) \\ f_{2N+2} = -s_1 & t \in [\tau_N + t_2, p(\bar{\tau})) \end{cases} \tag{13}$$

The cost function, $L(x)$, is set in order to maximize the quality of the bridge. Therefore, let $L(x) = -x$, i.e, minimize the negative quality. The cost on repairs should be decreasing for a longer time in between (e.g more interest on existing funds) and so the cost function g is set to $g(z) = \frac{\alpha}{z+\beta}$. The positive constant α is set to weight the importance of bridge quality versus the advantage of waiting with a repair, while also putting an actual cost on every repair. β, also a positive constant (small), is set to avoid the cost to blow up as switches merge together. The cost functional is then expressed as

$$J(\bar{\tau}) = \int_{t_0}^{p(\bar{\tau})} -x dt + \sum_{j=1}^{N} \frac{\alpha}{z_j + \beta} \tag{14}$$

where

$$z_j = \tau_j - \tau_{j-1}$$

The bridge reparation optimizing problem, P_b, becomes

$$\min_{\bar{\tau}} J(\bar{\tau})$$

s.t.

$$\begin{cases} \dot{x} = f(t) \\ x(0) = x_0 \end{cases}$$

and

$$t_0 = \tau_0 \leq \tau_0 + t_1 \leq \tau_1 \leq \tau_1 + t_2 \leq \dots$$
$$\leq \tau_{n-1} + t_2 \leq \tau_N \leq \tau_N + t_2 \leq \tau_{N+1} = T \tag{15}$$

and as developed in section II, the equations for the costate becomes

$$\begin{cases} \lambda(T) = 0 \\ \dot{\lambda} = 1 \end{cases}$$
$$\forall t \in [0, p(\bar{\tau})]. \tag{16}$$

Due to the alteration of the dynamics as a result of the times t_2, the optimization of the switching times need to take consideration of the partial derivatives $\frac{dJ}{d(\tau_i+t_2)}$. Although, the times $\tau_i + t_2$ are not moveable themselves but dependent on the respective τ_i, they still are switches from a switched system point of view. Hence, the partial derivatives have to be altered according to

$$\begin{aligned} \frac{dJ}{d\tau_i} &= \frac{\partial J}{\partial \tau_i} + \frac{\partial J}{\partial(\tau_{i,2})} \\ \tau_{i,2} &= \tau_i + t_2 \end{aligned} \tag{17}$$

and the resulting derivative becomes

$$\begin{aligned} \frac{dJ}{d\tau_i} = {} & \lambda^T(\tau_i)\big[f_{2i}(x(\tau_i)) - f_{2i+1}(x(\tau_i))\big] \\ & + \lambda^T(\tau_i + t_2)\big[f_{2i+1}(x(\tau_i + t_2)) - f_{2i+2}(x(\tau_i + t_2))\big] \\ & + \frac{\partial g_i}{\partial z_i} - \frac{\partial g_{i+1}}{\partial z_{i+1}} \end{aligned} \tag{18}$$

where g_{N+1} represents the cost between the final repair and the complete rebuild of the bridge. Note that the term for the final time constraint in the derivative, disappears as the final state is $x = 0$.

5.1.1 Solution

Assume that the bridge whose repairs is to be optimized will have a total of three switches between being built and totally rebuilt. Assume furthermore that the quality of the newly built bridge is $x_0 = 5$ and the two slopes $s_1 = 0.1, s_2 = 0.01$. Let the times where the degeneration is slowed down due to repairs and the bridge being new be $t_1 = 15, t_2 = 10$ years. To weight the importance between the level of the quality and the cost on/between repairs, the parameters of $g(z_i)$ is set to $\alpha = 250$ and $\beta = 0.01$. The bridge related parameters, x_0, s_1, s_2, t_1 and t_2 were provided by [3].

For the numerical computation, the Armijo stepsize is, as generally, set to $\alpha = \beta = 0.5$. Time is discretized to time steps of $dt = 0.01$ and the initial switching vector (excluding initial and final time) is chosen as $\bar{\tau} = [\tau_1 \ \tau_2 \ \tau_3] = [30 \ 45 \ 65]$. Figure 8 compares the trajectory with optimized repairs (blue) to the one with the initial repairs (dotted red). The algorithm terminates when the maximum of desired step lengths for the different switching times is less than the discretized time steps. The resulting switching vector is found to be

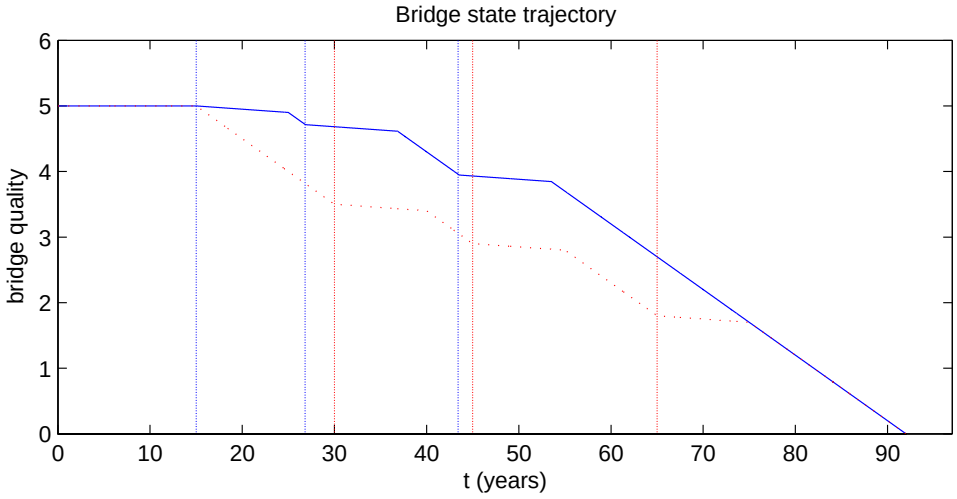

Figure 8: Bridge repair optimization: Optimized quality trajectory vs initial

[15.0 26.8 43.4]. The cost functional, $J(\bar{\tau})$, and the norm of the projected gradient, $h(\bar{\tau})$, for every iteration i are shown in fig 9.

Proposition 5.1 *The solution to problem P_b is in fact a global minimum.*

Proof 5.1 *Divide the cost functional into two $J = J_1 + J_2$ where*

$$J_1 = \int_{t_0}^{p(\bar{\tau})} -x dt$$
$$J_2 = \sum_{j=1}^{3} \frac{\alpha}{z_j + \beta}.$$

The partial derivatives of the first cost functional J_1 is found to be

$$\frac{\partial J_1}{\partial \tau_1} = \frac{\partial J_1}{\partial \tau_2} = \frac{\partial J_1}{\partial \tau_3} = s_1 t_2 - s_2 t_2 \tag{19}$$

and so J_1 is linear with all τ_i's. The functional J_2 only comprise terms of type $\frac{1}{z_i+\beta}$ which are known to be convex, since both the nominator and denominator is positive ($\alpha, \beta, z_i \geq 0$).

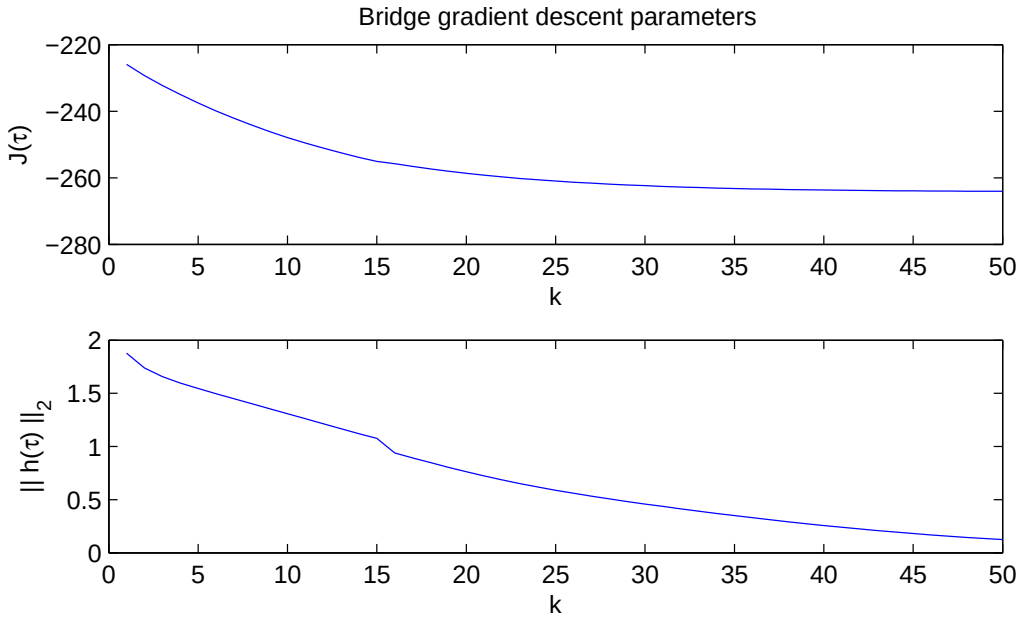

Figure 9: Bridge repair optimization: Cost and norm of the projected gradient for every iteration

The two cost functionals are then linear and convex functions and hence, the sum of them, J, is a convex function.

■

The partial derivatives in eq. 19 was developed with the symbolic toolbox in matlab. The code is found in appendix B.

The model proposed was confirmed to be appropriate by the department of Civil Engineering at UCB [3]. Important to know, is that the selection of certain parameters is very subjective. To illustrate how dependent this model is of the weighting constant α, weight between bridge quality and benefits of waiting with a repair, figure 10 shows cost dependency of the number of switches for different values of α. Notice, that for α small the optimal number of switches is highest possible and the opposite for α large. With the current parameters on the bridge model, the maximum possible repairs is 50. Also note that the functions are convex and so there is a optimum number of switches depending on chosen α.

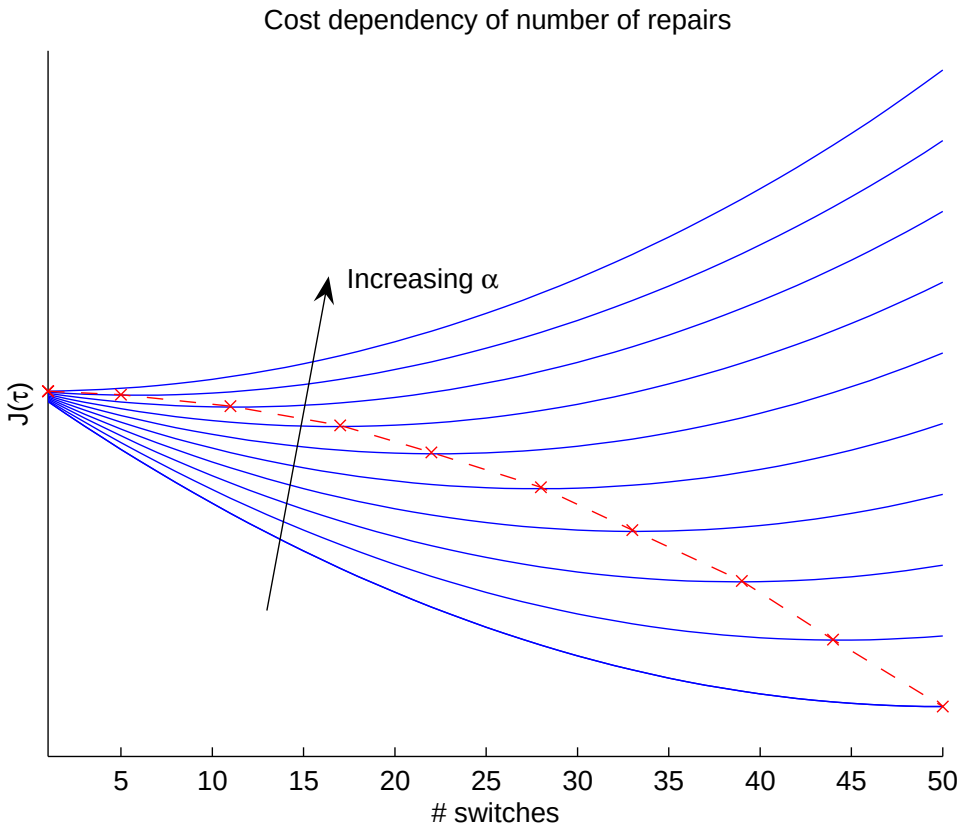

Figure 10: The cost as a function of number of repairs for different values of α. The red dotted line marks the optimum number of repairs.

CHAPTER VI

LIMITATIONS

The final time T in the proceeding bridge repairs optimization problem occurred when the state reached zero. As a result, the cost function also reached zero for the final time $L(x(T)) = 0$ and hence, the partial derivative of the constraint, $\frac{\partial p}{\partial \tau_i}$, was eliminated in eq. 7 for the cost functional derivative. Accordingly, having a constraint $x(T) = 0$ simplifies the problem as of not having to deal with expressing the constraint $T = p(\bar{\tau})$ at all. On the other hand when dealing with numerics and simulation, constraints of type $x(T) = x_T$ is easy to handle as telling when to stop simulating the trajectory forward.

The algorithm proposed naturally reduces the number of switches when the first switch, τ_1, is pushed to t_0 and/or when the last, τ_N, is pushed to T. It also reduces the number when two adjacent switching times, τ_i and τ_{i+1} merges to one single switching point. In the bridge example, a maximum number of switches existed. If the case is not so, this algorithm does not decide whether or not insertions of new switches would be beneficial.

CHAPTER VII

CONCLUSION

Previous work in the area of optimal control of switched systems has developed tools for finding optimal switching times given a cost functional defined on the systems trajectory. Inspired by an existing problem concerning optimizing bridge repairs, this thesis investigated the consequence of extending this cost functional with explicit costs on the switches and/or in between switches and having the final time given by a constraint. A solution for the gradient of the cost functional was derived and an existing algorithm was proposed to be used for numerical solutions.

As expected, the results herein derived with a calculus of variations approach, are very similar to the results from previous work that it is based on. The equation for the costate is identical to the one used in other research. Furthermore, the gradient only has some extra terms related to the extensions. The terms related to the costs on and/or between switches are very straightforward to handle and impose only derivatives of the introduced cost functions. The derivative related to the final time constraint impose no inconvenience if, as in the previous bridge problem, the cost function L of the state at the final time is equal to zero. However, if it is not, this might impose difficulties to calculate the derivatives of the constraint, as the constraint can be hard to explicitly express in terms of the dynamics and switching times.

A steepest-descent algorithm with gradient projection and Armijo stepsizes, presented in previous work, was proposed to be used for the problem to achieve feasible numerical results. The algorithm has been proven to globally converge to stationary points. Concerning the bridge, the gradient formula and the algorithm was used in a matlab implementation to reach a global minimum when optimizing the repairs on it.

APPENDIX A

MATLAB IMPLEMENTATION

A.0.2 switch_cost.m

```
%clear

%Define dynamics in f.m
%Define costate dynamics in dlambdadt.m
%Define gradient in dJdtao.m
%Define final lambda in lambdaT.m
%Define cost in J.m
%Define constraint in const.m

global X;
global TT;
global LAMBDA;
global tao;
global ts;
global t0;
global T;
global dt;
global x0;
global alfag;
global betag;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameters for the bridge problem %%
N=50; %Max iterations
alfa=0.5; %Armijo constant
beta=0.5; %Armijo constant

x0=5; %Start value
t0=0; %Start time
dt=0.1; %Time step length

alfag=250;
betag=0.01;

%%1 Pick arbitrary taos
tao=[30 45 65];
ts=[15 10 10 10];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TAO=[tao']; %Contains taos (for plotting)
ndJ=[]; %Contains gradients (for plotting)
H=[]; %Contains proj. gradient norms (for plotting)
JJ=[]; %Contains costs (for plotting)
Z=[]; %Contains steplengths (for plotting)

for (it=1:N)
```

```
X=[]; %Contains states
TT=[]; %Contains times
LAMBDA=[]; %Contains costates
dJ=[]; %Contains gradients
x=x0;
X=[X,x];
t=t0;
TT=[TT;t];
tao=round(tao./dt)*dt; %Make taos have values out of TT
modes=[t0+ts(1)];
for(i=1:length(tao))
    modes=[modes,tao(i),tao(i)+ts(i+1)];
end

%%2 Solve x forwards
while (x>0)
    mode=find(modes>t,1);
    if (isempty(mode==0))
        mode=length(modes)+1;
    end
    x=x+dt.*f(x,mode);
    X=[X,x];
    t=t+dt;
    TT=[TT;t];
end

T=t;

if (it==1)  %For plotting initial trajectory
    X0=X;
    TT0=TT;
end


%%3 Solve lambda backwards
t=T;
lambda=lambdaT();
LAMBDA=[lambda;LAMBDA];

while (t≥t0)
    x=X(:,find(TT≥t,1));
    mode=find(modes>t,1);
    if (isempty(mode==0))
        mode=length(tao)+1;
    end
    lambda=lambda-dt.*dlambdadt(x,lambda,mode);
    LAMBDA=[lambda;LAMBDA];
    t=t-dt;
end


%%4 Update tao
dJ=dJdtao();

%Assign blocks
taob=[1];
for (i=2:length(tao))
    if (tao(i)==tao(i-1)+ts(i))
```

```
            taob(i)=taob(i-1);
        else
            taob(i)=taob(i-1)+1;
        end
    end
    %Compute all r
    R=zeros(length(tao));
    for (i=1:length(R))
        for (j=i:length(R))
            R(i,j)=1/(j-i+1)*sum(dJ(i:j));
        end
    end
    %Compute h
    h=zeros(1,length(tao));
    for (i=1:taob(length(taob)))
        k=find(taob==i,1);
        l=k;
        n=find(taob==i,1,'last');
        while (l≠n+1)
            rmax=max(R(l,l:n));
            m=find(R(l,l:n)==rmax,1,'last')+l-1;
            for (j=l:m)
                h(j)=-rmax;
                if ((tao(j)==t0+ts(1) & rmax>0) | (tao(j)+ts(j+1)==T
                    & rmax<0))
                    h(j)=0;
                end
                if (j>1 & taob(j)==taob(j-1) & h(j-1)==0)
                    h(j)=0;
                end
            end
            l=m+1;
        end
    end
    %Check constraint
    zmax=1;
    while (¬const(tao+zmax.*h))
        zmax=zmax-0.001;
        zmax=round(zmax/0.001)*0.001;
    end
    z=zmax;
    k=1;
    %Make sure J decreases
    while (¬((J(tao+z.*h)-J(tao))≤(alfa*z*h*dJ')))
        z=zmax*beta^k;
        k=k+1;
        if (z<1e-7)
            z=0;
        end
    end

    JJ=[JJ,J(tao)];

    ndJ=[ndJ,abs(dJ)'];
    H=[H,norm(h)];
    Z=[Z,z];

    if (z*max(abs(h))<dt)
        break
```

```
    end

    tao=tao+z.*h;
    TAO=[TAO,tao'];

    %Back to 2
end

%% Results for the bridge
figure(1);
subplot(2,1,1);
plot(JJ,'-');
ylabel('J(\tau)');
xlabel('k');
subplot(2,1,2);
plot(H,'-');
ylabel('||␣h(\tau)␣||_{2}');
xlabel('k');

figure(2);
plot(TT,X,TT,0,':k',TT0,X0,':r');
hold on;
for(i=1:length(tao))
    plot(TAO(i,1),TT0,':r');
    plot(TAO(i,size(TAO,2)),TT,':b');
end
xlim([0 T+5]);
ylim([0 x0+1]);

tao
```

A.0.3 const.m

```
function [a] = const(tao)

%Checks the constraint t0≤tao1≤...≤T

global t0;
global T;
global ts;
a=1;

%% Constarint for the bridge problem %%
for (i=1:length(tao)-1)
    if (tao(i)+ts(i+1)>tao(i+1))
        a=0;
    end
end
if ((tao(1)<t0+ts(1)) | (tao(length(tao))+ts(length(tao)+1)>T))
    a=0;
end
```

A.0.4 dJdtao.m

```
function [dJ] = dJdtao()

%returns dJdtao=[dJdtao1 dJdtao2 ...]

global X;
global TT;
global LAMBDA;
global tao;
global t0;
global T;
global ts;
global alfag;
global betag;

dJ=[];

for (j=1:length(tao))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Cost derivative for the bridge problem %%
    if(length(tao)>1)
        if (j==1)
            zii=tao(j)-t0;
            zi=tao(j+1)-tao(j);
        elseif (j==length(tao))
            zii=tao(j)-tao(j-1);
            zi=T-tao(j);
        else
            zii=tao(j)-tao(j-1);
            zi=tao(j+1)-tao(j);
        end
    else
        zii=tao(j)-t0;
        zi=T-tao(j);
    end
    xtao=X(:,find(TT>=tao(j),1));
    lambdatao=LAMBDA(find(TT>=tao(j),1),:);
    dJ(j)=lambdatao*(f(xtao,2*j)-f(xtao,2*j+1))+alfag*(-1/(zii+betag)^2)
            -alfag*(-1/(zi+betag)^2);
    xtao=X(:,find(TT>=tao(j)+ts(j+1),1));
    lambdatao=LAMBDA(find(TT>=tao(j)+ts(j+1),1),:);
    if(j==14)

    end
    dJ(j)=dJ(j)+lambdatao*(f(xtao,2*j+1)-f(xtao,2*j+2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

A.0.5 dlambdadt.m

```
function [dlambdadt] = dlambdadt(x,lambda,mode)

%Returns dlambdadt dep. on interval (mode)

%% Costate diff. eq. for the bridge problem %%
dlambdadt=1;
```

A.0.6 f.m

```
function [f] = f(x,mode)

%Returns f1(x),f2(x),... dep. on interval (mode)

%% Dynamics for the bridge problem %%
switch mod(mode,2)
    case {0}
        f=-0.1;
    case {1}
        f=-0.01;
end

if(mode==1)
    f=0;
end
```

A.0.7 J.m

```
function [J] = J(tao)

%Calculates the cost for a given set of taos

global T;
global t0;
global dt;
global ts;
global x0;
global alfag;
global betag;

x=x0;
t=t0;
J=0;

modes=[t0+ts(1)];
for(i=1:length(tao))
    modes=[modes,tao(i),tao(i)+ts(i+1)];
end

while (t<=T)
    mode=find(modes>t,1);
    if (isempty(mode==0))
        mode=length(modes)+1;
    end
    x=x+dt.*f(x,mode);
    t=t+dt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Cost for the bridge problem %%
    J=J+dt*(-x);
end
for (i=2:length(tao))
    J=J+alfag/(tao(i)-tao(i-1)+betag);
```

```
end
J=J+alfag/(tao(1)-t0+betag)+alfag/(T-tao(length(tao))+betag);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.0.8 lambdaT.m

```
function [lambdaT] = lambdaT()

%Returns final lambda

%% Final costate for the bridge problem %%

lambdaT=0;
```

APPENDIX B

GLOBAL OPTIMUM CODE

```
clear
syms x1 x2 x3 x4 x5 x6 x0 s1 s2 t1 t2 tau1 tau2 tau3
            t T x alfa beta;
x1=x0;
x2=x0+s1*t1-s1*t;
x3=x0+s1*t1-s1*tau1+s2*tau1-s2*t;
x4=x0+s1*t1-s1*tau1+s2*tau1-s2*(tau1+t2)+s1*(tau1+t2)-s1*t;
x5=x0+s1*t1-s1*tau1+s2*tau1-s2*(tau1+t2)+s1*(tau1+t2)-s1*tau2
    +s2*tau2-s2*t;
x6=x0+s1*t1-s1*tau1+s2*tau1-s2*(tau1+t2)+s1*(tau1+t2)-s1*tau2
    +s2*tau2-s2*(tau2+t2)+s1*(tau2+t2)-s1*t;
x7=x0+s1*t1-s1*tau1+s2*tau1-s2*(tau1+t2)+s1*(tau1+t2)-s1*tau2
    +s2*tau2-s2*(tau2+t2)+s1*(tau2+t2)-s1*tau3+s2*(tau3)-s2*t;
x8=x0+s1*t1-s1*tau1+s2*tau1-s2*(tau1+t2)+s1*(tau1+t2)-s1*tau2
    +s2*tau2-s2*(tau2+t2)+s1*(tau2+t2)-s1*tau3+s2*(tau3)
    -s2*(tau3+t2)+s1*(tau3+t2)-s1*t;
J=int(-x1,t,0,t1)+int(-x2,t,t1,tau1)+int(-x3,t,tau1,tau1+t2)
    +int(-x4,t,tau1+t2,tau2)+int(-x5,t,tau2,tau2+t2)
    +int(-x6,t,tau2+t2,tau3)+int(-x7,t,tau3,tau3+t2)
    +int(-x8,t,tau3+t2,T);
dJdtau1=diff(J,tau1)
dJdtau2=diff(J,tau2)
dJdtau3=diff(J,tau3)
```

REFERENCES

[1] Armijo, L., "Minimization of functions having lipschitz continuous first-partial derivatives," *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[2] Axelsson, H., Egerstedt, M., Wardi, Y., and Vachtsevanos, G., "Algorithm for switching-time optimization in hybrid dynamical systems," in *Proc. Mediterranean Conference on Control and Automation*, (Limassol, Cyprus), June 2005.

[3] Bayen, A. M. private communication, Oct. 2005.

[4] Branicky, M., Borkar, V., and Mitter, S., "A unified framework for hybrid control: Model and optimal control theory," vol. 43, pp. 31–35, Jan. 1998.

[5] Brockett, R., "Stabilization of motor networks," in *Proc. 34th IEEE Conference on Decision and Control*, (New Orleans, Louisiana), pp. 1484–1488, Dec. 1995.

[6] Decarlo, R., Branicky, M., Pettersson, S., and Lennartson, B., "Perspectives and results on the stability and stabilizability of hybrid systems," vol. 88, pp. 1069–1082, July 2000.

[7] Egerstedt, M. and Wardi, Y., "Multi-process control using queuing theory," in *Proc. 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 1991–1996, Dec. 2002.

[8] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched systems," submitted for publication.

[9] Egerstedt, M., Wardi, Y., and Delmotte, F., "Optimal control of switching times in switched dynamical systems," in *Proc. 42nd IEEE Conference on Decision and Control*, (Maui, Hawaii), pp. 2138–2143, Dec. 2003.

[10] Giua, A., Seatzu, C., and der Mee, C. V., "Optimal control of switched autonomous linear systems," in *Proc. 40th IEEE Conference on Decision and Control*, (Orlando, Florida), pp. 2472–2477, Dec. 2001.

[11] Hedlund, S. and Rantzer, A., "Optimal control of hybrid systems," in *Proc. 38th IEEE Conference on Decision and Control*, (Phoenix, Arizona), pp. 3972–3977, Dec. 1999.

[12] Hristu-Varsakelis, D., "Feedback control systems as users of shared network: Communication sequences that guarantee stability," in *Proc. 40th IEEE Conference on Decision and Control*, (Orlando, Florida), pp. 3631–3636, Dec. 2001.

[13] Lennartson, B., Tittus, M., Egardt, B., and Pettersson, S., "Hybrid systems in process control," vol. 16, pp. 45–56, Oct. 1996.

[14] Lincoln, B. and Rantzer, A., "Optimizing linear systems switching," in *Proc. 40th IEEE Conference on Decision and Control*, (Orlando, Florida), pp. 2063–2068, Dec. 2001.

[15] Lincoln, B. and Rantzer, A., "Relaxed optimal control of piecewise linear systems," in *Proc. IFAC Conference on Analysis Design of Hybrid Systems*, (St. Malo, France), June 2003.

[16] Polak, E., *Optimization Algorithms and Consistent Approximations*. New York, New York: Springer-Verlag, 1997.

[17] Rehbinder, H. and Sanfirdson, M., "Scheduling of a limited communication channel for optimal control," in *Proc. 39th IEEE Conference on Decision and Control*, (Sidney, Australia), pp. 1011–1016, Dec. 2000.

[18] Shaikh, M. and Caines, P., "On trajectory optimization for hybrid systems: Theory and algorithms for fixed schedules," in *Proc. 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 1997–1998, Dec. 2002.

[19] Shaikh, M. and Caines, P., "On the optimal control of hybrid systems: Optimization of trajectories, switching times and location schedules," in *Proc. 6th Intl. Workshop on Hybrid Systems: Computation and Control*, (Prague, Czech Republic), pp. 466–481, Apr. 2003.

[20] Tavernini, L., "Differential automata and their discrete simulators," *Nonlinear Anal. Theory, Methods, Appl.*, vol. 11, pp. 665–683, no. 6 1987.

[21] Walsh, G., Ye, H., and Bushnell, L., "Stability analysis of networked control systems," in *Proc. American Control Conference*, (San Diego, California), pp. 2876–2880, June 1999.

[22] Witsenhausen, H., "A class of hybrid-state continuous dynamic systems," vol. AC-11, pp. 161–167, no. 2 1966.

[23] Xu, X. and Antsaklis, P., "Optimal control of switched autonomous systems," in *Proc. 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 4401–4406, Dec. 2002.

[24] Xu, X. and Antsaklis, P., "Optimal control of switched systems via nonlinear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75, pp. 1406–1426, Nov. 2002.

[25] Xu, X. and Antsaklis, P., "An approach to switched systems optimal control based on parameterization of the switching instants," vol. 49, pp. 2–16, Jan. 2004.